User Manual

MSC STK-TDC-Mach300

Rev. 1.0

Date: 2011/07/28

MSC Vertriebs GmbH
Industriestraße 16
76297 Stutensee
Germany

Author: AP, AKL
Phone: +49 7249 910 288
Fax: +49 7249 910 4388
Email: AP@msc-ge.com

© MSC. All rights reserved. Although great care has been taken in preparing this document, MSC can not be held responsible for any errors or omissions. All information in here is subject to change without notice. All hardware and software names used are trade names and/or trademarks of the respective owners.

Contents

1 Introduction 5
2 Installing the hardware 6
3 Installing and running the software 7
3.1 Host software installation 7
3.2 TDC starter kit measurement software overview 8
3.3 Application software functions 9
3.3.1 Connect/Disconnect button 9
3.3.2 I/O history button 9
3.3.3 I/O protocol checkbox 9
3.3.4 Hardware setting checkboxes 9
3.3.5 TDC reset button 10
3.3.6 User defined function buttons 10
3.3.7 Exit button 10
3.3.8 Input line 10
3.3.9 Protocol box 10
3.3.10 Clear protocol button 11
3.4 TDC starter kit measurement software commands 11
3.5 User defined functions 12
3.5.1 Channel resolution 12
3.5.2 Single shot 12
3.5.3 N-times single shot 13
4 Creating your own measurements programs 14
4.1 Software requirements 14
4.2 Create your own C# project 14
4.3 Library MSCSTK-LIB 14
4.4 Function calls from MSCSTK-LIB 15
4.4.1 Communication with the starter kit 15
4.4.2 Using the protocol box 16
4.4.3 Initialisation the hardware and control the behaviour of the MSCSTKForm form elements . 17
4.4.4 Working with the `OptionDialog` form 19
4.5 Implementing your program into the GUI 19

5 Appendix 21
5.1 Specification 21
5.1.1 Common 21
5.1.2 Minimum system requirements 21
5.1.3 Power supply 21
5.1.4 Reset 21
5.2 Layout diagrams / schematics 22

Figures

Figure 1: Evaluation board block diagram 5
Figure 2: Connect the starter kit 6
Figure 3: MSCSTKTDC-Installer software 7
Figure 4: Start up screen of the starter kit host software 7
Figure 5: Host software connected with a TDC-Mach300 starter kit 8
Figure 6: Select options dialog 14
Figure 7: Layout diagram TDC-Mach300 starter kit (top view) 22
Figure 8: Schematic of the TDC-Mach300 starter kit (part 1) 23
Figure 9: Schematic of the TDC-Mach300 starter kit (part 2) 24

Tables

Table 1: I/O history line description 9
Table 2: Additional input line control commands 10
Table 3: Protocol box text types 10
Table 4: TDC-Mach300 opcodes for input line or user defined functions 11
Table 5: TDC-Mach300 hardware specific opcodes for input line or user defined functions 12
Table 6: MSCSTK-LIB defined forms 14
Table 7: MSCSTK-LIB defined form elements and variables 15
Table 8: function Cmd() 15
Table 9: variable Result[] 16
Table 10: variable RunProgram 16

tdc@msc-ge.com www.msc-ge.com

Table 11: function WriteText() 16
Table 12: function WriteDebug() 16
Table 13: variable Color 16
Table 14: variable DebugMode 16
Table 15: function InitTDC() 17
Table 16: function RestorePinSettings() 17
Table 17: function SavePinSettings() 17
Table 18: function Wait() 17
Table 19: variable AutoConnectButton 17
Table 20: variable VersionLabel 17
Table 21: variable WorkingProgressBar 17
Table 22: variable CalClkPeriod 18
Table 23: variable GlobalTimeOutValue 18
Table 24: structure TDC 18
Table 25: function GenericOptionsDialog() 19
Table 26: function ShowConfiguration() 19

Examples

Example 1: I/O history 9
Example 2: Protocol box 11
Example 3: Place a program on user function button 9 20
Example 4: button 9 output in protocol box 20

1 Introduction

In combination with a PC the MSC STK-TDC-Mach300 starter kit is an easy to use evaluation tool for the MSC Time to Digital Converter TDC-Mach300. The starter kit consists of the evaluation board and the necessary accessories. Figure 1 shows the block diagram of the evaluation board. It consists of a microprocessor which controls the communication between the TDC and the PC using an USB 1.1-interface. Three SMB-connectors are provided for high-precision time difference measurements between the start- and two stop-inputs. With the RC-measuring ports resistance measurements can be executed. All important signals of the TDC chip are accessible for analyses (e.g.: logic analyser) via two connectors.

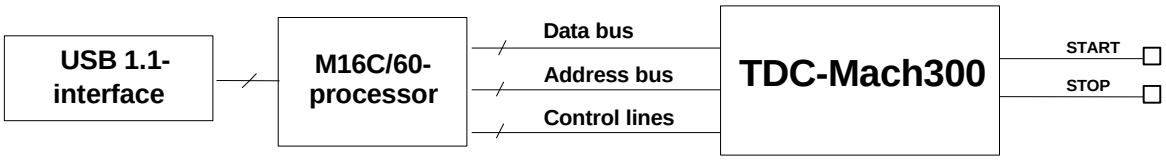


Figure 1: Evaluation board block diagram

The starter kit provides the following components:

- 1 power supply unit, I_{min} = 500 mA
- 1 evaluation board
- 1 CD including software and documentation
- 2 measurement cables
- 1 user manual

2 Installing the hardware

First of all the evaluation board has to be powered by connecting the provided power supply unit to the connector X201 (see figure 7). The red Power-LED (D201) turns on. After this the evaluation board has to be connected to an USB-port of the PC using an A/B USB-cable. The green Run-LED (D202) turns on and indicates the correct operation of the M16C-processor. The evaluation board is ready-to-operate.

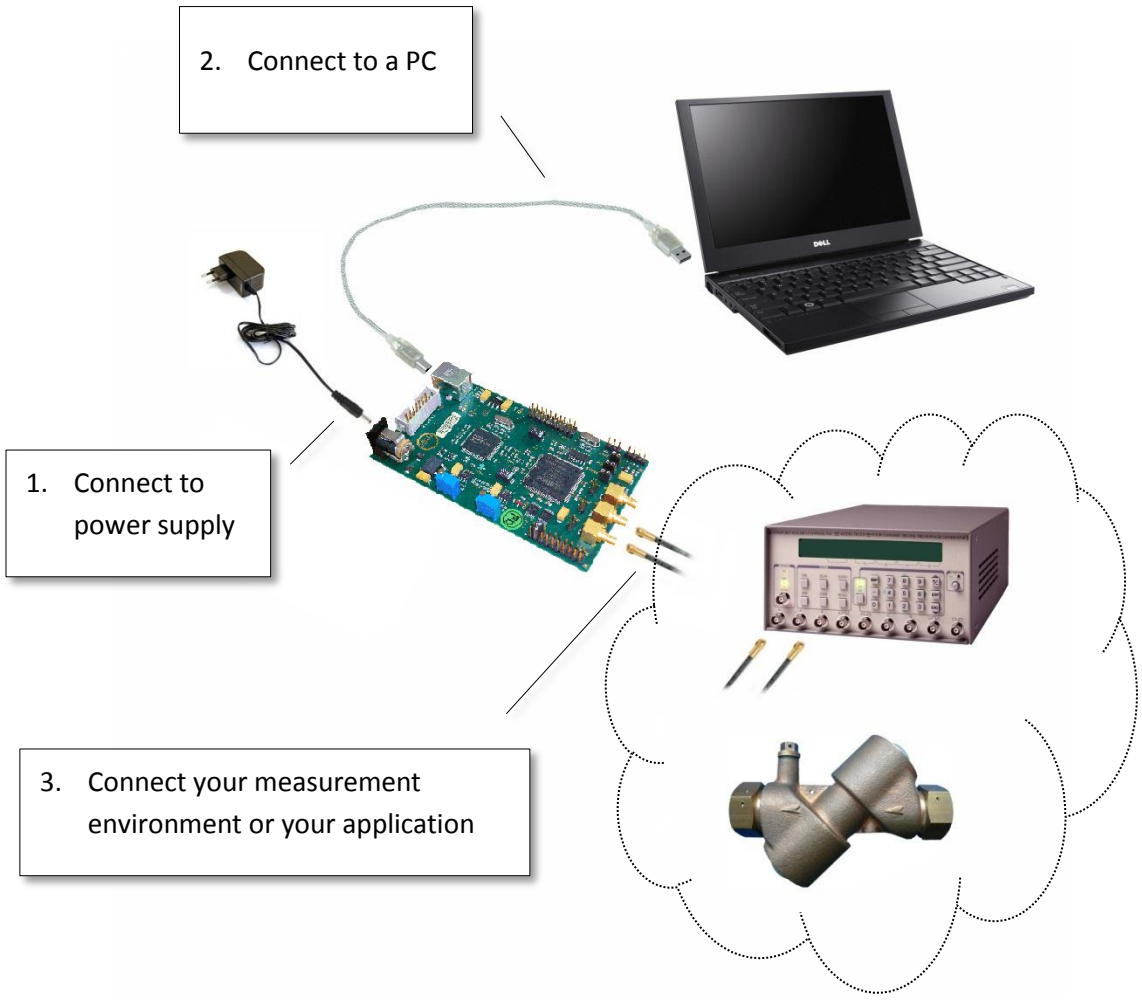


Figure 2: Connect the starter kit

3 Installing and running the software

3.1 Host software installation

The software is delivered on the starter kit CD. Driver software, libraries and source codes are provided. The starter kit software with driver and all provided sources can be installed by starting the installer executable “MSCSTKTDC-Installer.exe”. You must have administration rights. Follow the instructions of the install program. Afterwards the software is ready to operate.

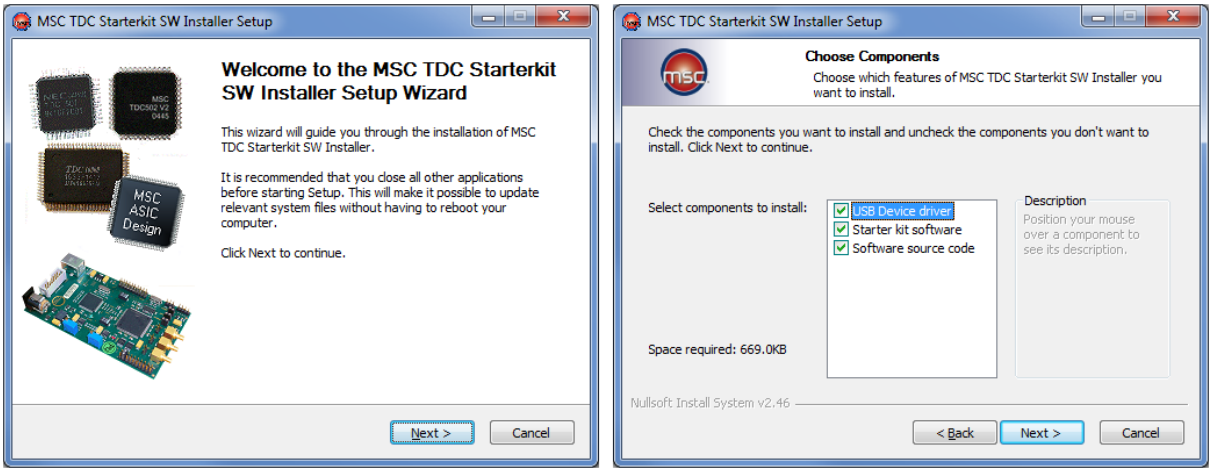

MSC TDC Starterkit SW Installer Setup
Welcome to the MSC TDC Starterkit SW Installer Setup Wizard
This wizard will guide you through the installation of MSC TDC Starterkit SW Installer.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel
MSC TDC Starterkit SW Installer Setup
Choose Components
Choose which features of MSC TDC Starterkit SW Installer you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
USB Device driver
Starter kit software
Software source code
Description
Position your mouse over a component to see its description.
Space required: 669.0KB
Nullsoft Install System v2.46
< Back
Next >
Cancel

Figure 3: MSCSTKTDC-Installer software

First of all ensure that the USB-driver ‘CDC_Demo.inf’ has been installed successfully. Otherwise power the starter kit, connect it to your PC and try to install the driver manually. The driver is on the CD.

The starter kit software is started by double-clicking on the file “TDC Starter Kit Measurement Software.exe”. The program’s application window appears.

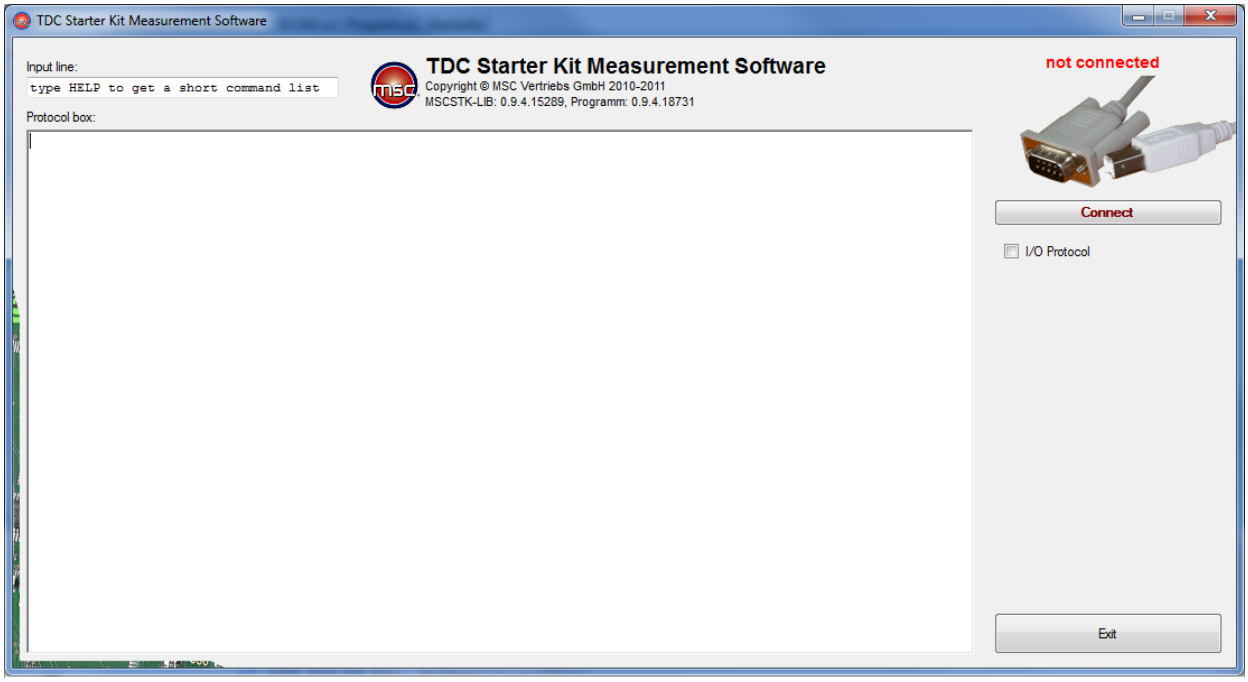


Figure 4: Start up screen of the starter kit host software

If the starter kit is powered and connected to the host, the software connects to the kit. Otherwise you can connect the starter kit to the software by pressing the “Connect”-button.

After successful connection the software shows you the connected kit. The green Run-LED turns off and the MSC STK-TDC-MACH300 application window sets up as shown in figure 5.

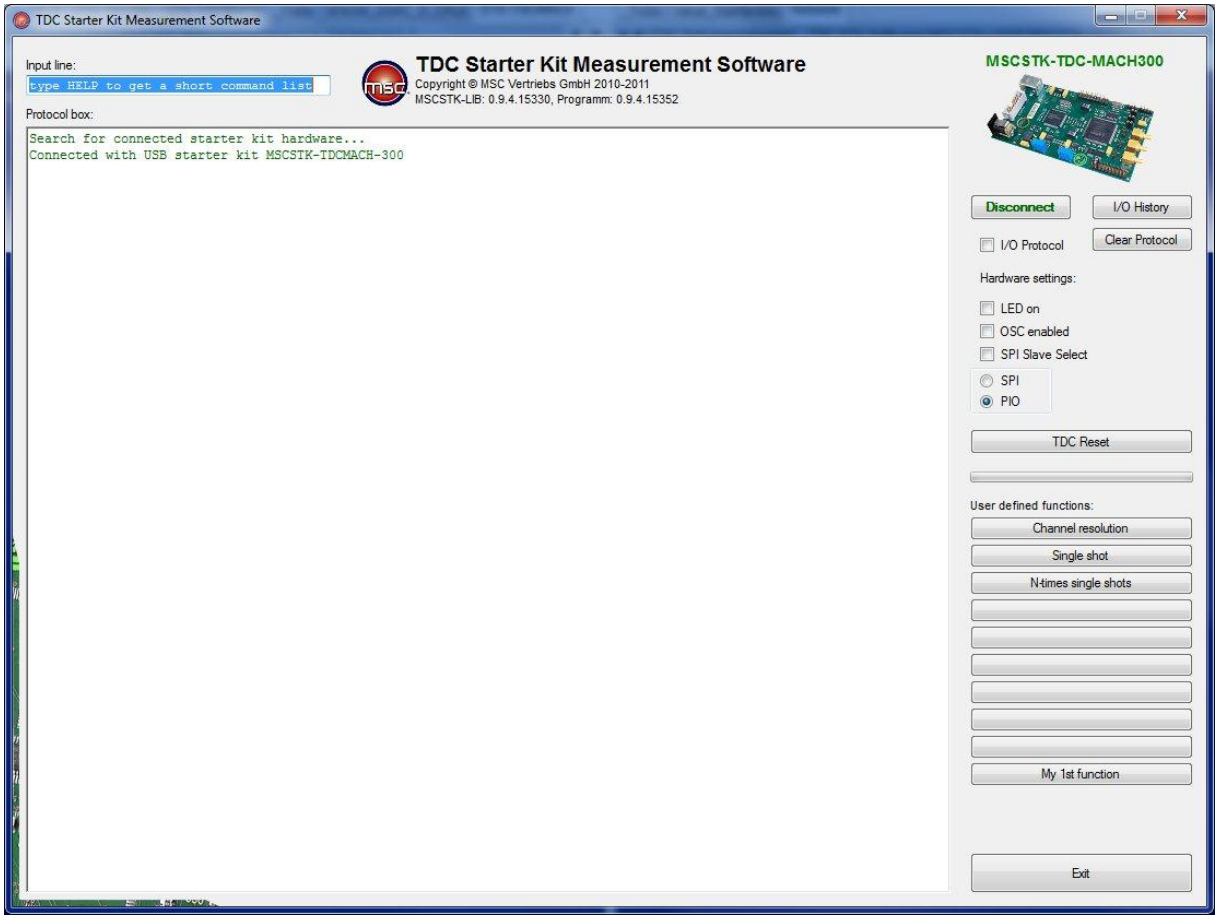


Figure 5: Host software connected with a TDC-Mach300 starter kit

3.2 TDC starter kit measurement software overview

The application window of the host software provides the following information and functions:

- Information about the connection status (upper right corner)
- Command input line on the top
- Protocol box for information, debug and dataflow outputs
- Program buttons and checkboxes on the right
- User defined function buttons on the right

3.3 Application software functions

3.3.1 Connect/Disconnect button

The “Connect”- button connects the software with the starter kit hardware. After the kit is connected, the software and the hardware are synchronised. The hardware setting checkboxes show the current status of the control pins and the button has changed its function to “Disconnect”.

3.3.2 I/O history button

The I/O history button shows a list of all transactions between the host software and the starter kit. Each entry has following information:

```
15:15:58,629 [0007]: COM8 G O [    0006] LED_OFF
```

15:15:58,629	**[0007]:**	**COM8**	**G**	**O**	**[0006]**	**LED_OFF**
Time stamp	List entry number	Com port	Command status: G Good F Fail	Data direction: I Input O Output ? Unknown	Command number or command reference	Transferred data

Table 1: I/O history line description

Example 1: I/O history

```
10:30:37,745 [0018]: COM8 G I [ACK 0011] #
10:30:37,753 [0019]: COM8 G O [    0012] WR_CTRL_REG 04
10:30:37,778 [0020]: COM8 G I [ACK 0012] #
10:30:37,783 [0021]: COM8 G O [    0013] RD_VALID
10:30:37,808 [0022]: COM8 G I [ACK 0013] #
```

This example shows the both commands “`WR_CTRL_REG 04`” and “`RD_VALID`” send from the host software to the starter kit and the received acknowledge “`#`” from the starter kit.

3.3.3 I/O protocol checkbox

If this checkbox is enabled, each data transfer between host software and starter kit is shown in the protocol box. Please consider that this function will slow down the application. It should be used only for debug purposes. Otherwise all data transfers could be displayed using the I/O history button.

3.3.4 Hardware setting checkboxes

The hardware setting checkboxes define the current status of the TDC control pins, the green Run-LED and the starter kit oscillator. E.g. ticking the ‘LED on’ checkbox will light the green Run-LED on the starter kit.

- ‘LED on’: Switches on and off the green Run-LED of the microprocessor.
- ‘OSC enable’: Enables and disables the TDC-calibration clock oscillator.
- ‘SPI Slave Select’: Enables (SPI slave selected) and disables (SPI in reset) the TDC’s SPI Slave Select pin SSN.
- ‘SPI’ / ‘PIO’: Communication between M16C-processor and TDC via SPI (TDC-pin SPI_EN = ‘1’) or via the parallel interface (SPI_EN = ‘0’).

Default settings after software connection are: LED off, oscillator disabled, SPI in reset (TDC-pin SSN = ‘1’) and parallel interface selected.

3.3.5 TDC reset button

The reset button resets the TDC by sending a low active reset pulse on the RSTN pin of the TDC. This button does not neither synchronise the software with the hardware nor manipulate any hardware setting.

Default setting after software connection is: TDC-reset pin inactive (TDC-pin RSTN = '1').

3.3.6 User defined function buttons

The user defined buttons are for the example programs, delivered with the software or appended by the starter kit user. Here you can add your own functionality.

3.3.7 Exit button

The exit button disconnects the starter kit and closes the application. All information like I/O history or protocol box will be lost.

3.3.8 Input line

The input line allows the user to control the TDC starter kit by input of commands described in chap. 3.4. Furthermore this line allows controlling some features of the host software:

Input line command	Function
`DEBUG_ON`	Enables debug outputs
`DEBUG_OFF`	Disables debug outputs
`CALCLKPERIOD value`	Change the calibration clock period to `value`
`GLOBALDEBUGLEVEL`	Filter the debug outputs by change the debug level between X and Y
`.`	Shows the internally stored values of the TDC structure
`HELP`	Show all available commands for the input line

Table 2: Additional input line control commands

The input line supports line editing and history scrolling function by the cursor keys.

3.3.9 Protocol box

The protocol box shows information of the software. The text types of the protocol box text can be:

Text type	Color	Text Type	Color
Titles	DarkCyan	Data input	DarkMagenta
Information	DarkGreen	Data output	Brown
Errors	Red	I/O history	Black
Debug	DarkGray	Help	MidnightBlue

Table 3: Protocol box text types

The protocol box is scrollable and supports select and copy functions. The copy supports RTF styles.

Example 2: Protocol box

```
Search for connected starter kit hardware...
Connected with USB starter kit MSCSTK-TDCMACH-300
> LED_ON
< #

Determination of the TDC resolution by 1 calibration measurements  [tCAL = 0,25 µs]

> OSC_EN
< #
> PU_RESET
< #
> SPI_RESET_ON
< #
```

3.3.10 Clear protocol button

The clear button clears the protocol box contents. Furthermore it clears the I/O- and commands history.

3.4 TDC starter kit measurement software commands

After connection with the starter kit hardware the host software allows the user to operate with the hardware. The user can input commands using the input line or can run a collection of commands written as a function for one of the user defined function buttons.

Each command has the following structure:

```
Opcode [Parameter]
```

To communicate with the starter kit the following list of opcodes is available for the input line and the program source code. Other data send to the starter kit are misinterpreted and result in an error message.

Opcode	Parameter	Description	Return value	Access mode
WR_CTRL_REG	8 Bit Hex	Control Register	--	wr
RD_CTRL_REG	--	Control Register	8 Bit Hex	rd
RD_STATUS_REG	--	Status Register	8 Bit Hex	rd
RD_CAL1_L	--	Calibration value CAL1: Low byte	8 Bit Hex	rd
RD_CAL1_H	--	Calibration value CAL1: High byte	8 Bit Hex	rd
RD_CAL2_L	--	Calibration value CAL2: Low byte	8 Bit Hex	rd
RD_CAL2_H	--	Calibration value CAL2: High byte	8 Bit Hex	rd
RD_VAL_L	--	Measurement value VAL: Low byte	8 Bit Hex	rd
RD_VAL_H	--	Measurement value VAL: High byte	8 Bit Hex	rd

Note: Pin EN_SPI = 0: Opcode-execution via parallel interface, EN_SPI = 1: Execution via SPI

Table 4: TDC-Mach300 opcodes for input line or user defined functions

Opcode	Parameter	Description	Return value (Hex)	Access Mode
RD_VALID	--	Reads TDC's pin VALID	0x00 or 0x01	rd
RD_OV	--	Reads TDC's pin OV	0x00 or 0x01	rd
MESS_MULTI	Number of measurements: 16 Bit Hex (max=0x07d0)	High-speed cycle of n single shot measurements with automatic calibration (tCAL= 250ns, tMIN= 50ns, tMAX= 750ns, auto noise; interface depends on pin EN_SPI)	n*6 Byte Hex: CAL1 (low, high), CAL2 (low, high), VAL (low, high)	wr/rd
INIT_SYSTEM	--	Message: STK-TDC-MACH300 is up !!!!	--	rd
RESET_ON	--	Puts the reset line to '0'	--	wr
RESET_OFF	--	Puts the reset line to '1'	--	wr
LED_ON	--	Switch-on M16C's green Run-LED	--	wr
LED_OFF	--	Switch-off M16C's green Run-LED	--	wr
OSC_EN	--	Enables TDC-calibration clock	--	wr
OSC_DIS	--	Disables TDC-calibration clock	--	wr
PU_RESET	--	Reset-pulse for the TDC on pin RSTN	--	wr
ENABLE_SPI	--	Puts the EN_SPI line to '1': SPI selected	--	wr
DISABLE_SPI	--	Puts the EN_SPI line to '0': Parallel Interface selected	--	wr
SPI_RESET_ON	--	Puts the SSN line to '1': SPI is in reset	--	wr
SPI_RESET_OFF	--	Puts the SSN line to '0': SPI Slave Select	--	wr
WRITE_SPI_0 [1]	--	Writes '0' to the MOSI line, reads one bit of the MISO line	0x00 or 0x01	wr/rd
WRITE_SPI_1 [1]	--	Writes '1' to the MOSI line, reads one bit of the MISO line	0x00 or 0x01	wr/rd

Table 5: TDC-Mach300 hardware specific opcodes for input line or user defined functions

3.5 User defined functions

On the right side of the application window you find ten 'User defined functions'-buttons. Some of them are predefined; all other are provided for creating your own measurement programs (see chap. 4).

3.5.1 Channel resolution

Clicking on the 'channel resolution'-button will find out the TDC's resolution using the interface, selected by the hardware settings PIO/SPI. Therefor the number of calibration measurements has to be entered within a pop-up window. The calibration values CAL1, CAL2 and the actual resolution of each measurement are displayed in the protocol box. At the end of the measurements the standard deviations of CAL1 and CAL2 and the average resolution are displayed. The calibration measurements are performed with a calibration clock period of t_{CAL} = 250ns.

3.5.2 Single shot

Clicking on the 'single shot'-button will start a single shot time difference measurement with automatic calibration using the interface, selected by the hardware settings PIO/SPI. The program waits for one rising edge on both the start-input START (connector X301) and the stop-input STOP (connector X303). The program executes a waiting loop. The waiting loop will be exited if either a start-stop event takes place

[1] Correct execution only when Pin EN_SPI = 1 and Pin SSN = 0

(correct execution) or a predefined waiting time has exceeded (error message). If a start-stop event takes place, the measurement result is calculated and displayed in the protocol box.

The minimum measurement period is t_{MIN} = 50ns (3.3V, 25°C) and the maximum measurement period is t_{MAX} = 750ns. The automatic calibration measurement is performed with a calibration clock period of t_{CAL} = 250ns.

Note: Remove jumpers JP301 and JP303 (see figure 7*) before the measurement because of accuracy reasons.*

3.5.3 N-times single shot

Clicking on the 'n-times single shots'-button will start a high-speed cycle of n single shot time difference measurements with automatic calibration and auto noise using the interface, selected by the hardware settings PIO/SPI. At first the number of measurements (≤ 2000) has to be entered within a pop-up window. Then the program awaits n start-stop measurements with rising edges on both the start-input START (connector X301) and the stop-input STOP (connector X303). Before each measurement the program executes a waiting loop. Each waiting loop will be exited if either a start-stop event takes place (correct execution) or a predefined waiting time has exceeded (error message). If all n start-stop events take place before the MESS_MULTI-command's timeout is reached (see table 8) all the n measurement results are calculated and displayed in the protocol box together with the minimum, the maximum and the average measurement time and the standard deviation. Otherwise an error message is generated.

The minimum measurement period of each measurement is t_{MIN} = 50ns and the maximum measurement period is t_{MAX} = 750ns. The automatic calibration measurements are performed with a calibration clock period of t_{CAL} = 250ns.

Note: Remove jumpers JP301 and JP303 (see figure 7*) before the measurements because of accuracy reasons.*

4 Creating your own measurements programs

4.1 Software requirements

To create your own measurement programs you have to fulfil the system requirements of the starter kit software and to install a C# development suite. In the following chapter we apply to the Microsoft Visual C# 2008 Express Edition.

4.2 Create your own C# project

First you have to set up a new project using the predefined template “Windows Form application”. Afterwards you have to add the necessary files to your project. First step is adding a link to the MSCSTK-LIB (mscstk-lib.dll) which is found in the source code directory of the install CD. Then add the existing elements “Program.cs”, “TDC300.cs”, “TDC330.cs”, “TDC501.cs”, “TDC502.cs”, “TDC10000.cs” and “UserButtons.cs” which can be found also in the source code directory of the install CD. Please overwrite the already “Program.cs” by the delivered one.

Now you should be able to compile the basic program for the first time. After starting the executable a window like figure 4 should appear. If this window does not appear, check if the file “Program.cs” initialises the right application form:

```
Application.Run(new MyMSCSTKForm());
```

The form1 which is initialised with the new Project is not necessary and may be deleted.

4.3 Library MSCSTK-LIB

To create your own test programs you have the complete language range of C# with all .NET- functions available. Additionally we have included some functions in our MSCSTK-LIB to enhance your programming success. This library has to be linked to your project. Otherwise you don’t have the main application form and all the functions to communicate with the starter kit. Following forms are available in MSCSTK-LIB:

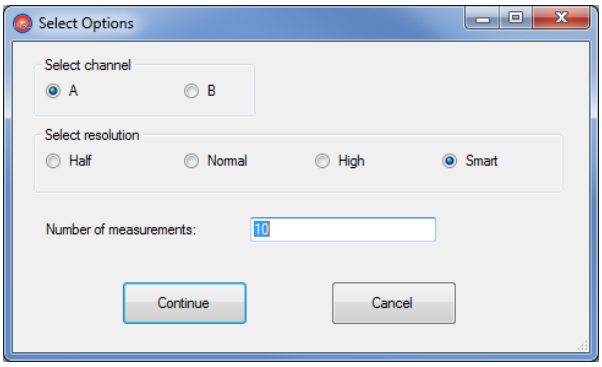
`MSCSTKForm`	Standard application form as shown in figure 5.
`OptionDialog`	Options dialog to set up various TDC or program parameters.  *Figure 6: Select options dialog*

Table 6: MSCSTK-LIB defined forms

A `TDCStruct` structure is implemented in MSCSTK-LIB. This structure contains the complete setting of the TDC and some program options.

Furthermore the following functions are available from MSCSTK-LIB:

- `int Cmd(string message, Int32 TimeOutValue)`
- `int Cmd(string message)`
- `void GenericOptionsDialog(bool OneChannel, bool Resolution, bool Measurements)`
- `void InitTDC(int Mode)`
- `void RestorePinSettings()`
- `void SavePinSettings()`
- `void ShowConfiguration(int Mode)`
- `void Wait(Int32 ms)`
- `void WriteDebug(string message)`
- `void WriteText(string message, Color color)`
- `void WriteText(string message, Color color, FontStyle style)`

At least a number of variables and all necessary form elements are marked as public.

Form elements	Global variables

`AutoConnectButton`
`VersionLabel`
`WorkingProcessBar`
	`double CalClkPeriod`
	`int GlobalTimeOutValue`
	`int[] Result`
	`bool DebugMode`
	`bool RunProgram`
	`Color DbgColor`
	`Color ErrColor`
	`Color HisColor`
	`Color InfColor`
	`Color InpColor`
	`Color OutColor`
	`Color TitColor`
	`Color HlpColor`
	`TDCStruct TDC`

Table 7: MSCSTK-LIB defined form elements and variables

4.4 Function calls from MSCSTK-LIB

4.4.1 Communication with the starter kit

Function call:	`int Cmd(string command [, Int32 TimeOutValue])`	
Description:	The function `Cmd` sends the `command` to the starter kit and receives the reply. The function call without a `TimeOutValue` uses the `GlobalTimeOutValue`. The time out value is used for the port communication.	
Parameter:	`command`	a valid opcode, optionally with parameters (see table 4 and table 5)
	`TimeOutValue`	Time out value in ms
Return value:	`-2`	The function was not executed, because the global variable `RunProgram` was `false`.
	`-1`	The function fails. Usually there is an error description message in the protocol box.
	`others`	The function was executed successfully. The number of results which are stored in the integer array variable `Result` is returned.

Table 8: function Cmd()

Variable:	`int[] Result`
Description:	This array contains the result values from the last successfully executed command. Attention: Only as many result values as returned from the function `Cmd` are updated!

Table 9: variable Result[]

Variable:	`bool RunProgram`
Description:	This variable is used to control the function call executions. If it turns to `false`, a function call was unsuccessfully and all further function calls of the MSCSTK-LIB will not be executed.

Table 10: variable RunProgram

4.4.2 Using the protocol box

Function call:	`void WriteText(string message, Color color [, FontStyle style])`	
Description:	The function `WriteText` outputs the string `message` to the protocol box. The text color `color` is necessary; using a font style is optionally.	
Parameter:	`message`	Output string
	`color`	Color setting for the output text
	`style`	Font style setting for the output text
Return value:	n/a	

Table 11: function WriteText()

Function call:	`void WriteDebug(string message)`	
Description:	The function `WriteDebug` outputs the string `message` to the protocol box.	
Parameter:	`message`	Output string
Return value:	n/a	

Table 12: function WriteDebug()

Variable:	`Color Color`		
Description:	The colour variables are used as the predefined colours for the output messages on the protocol box. The values are changeable. Predefined are following colours:		
	`DbgColor`	`DarkGray`	Debug outputs
	`ErrColor`	`Red`	Error messages
	`HisColor`	`Black`	History table colour
	`InfColor`	`DarkGreen`	Normal (information) text outputs
	`InpColor`	`DarkMagenta`	Colour used for port input messages
	`OutColor`	`Brown`	Colour used for port output messages
	`TitColor`	`DarkCyan`	Title colour

Example:	`This line was written as a additonal text fo debug features usign the colour "DbgColor".`
	`If an Error occurs you will see this with the colour "ErrColor".`
	`The normal text output is done usign the colour "InfColor".`
	`If you press the history button, the history is printed with "HisColor".`
	`Usually the title drawed usign "TitColor" is on the top of your measurement application.`		

Table 13: variable Color

Variable:	`bool DebugMode`
Description:	This variable is to enable you a debug mode, e.g. for additional outputs.

Table 14: variable DebugMode

4.4.3 Initialisation the hardware and control the behaviour of the MSCSTKForm form elements

Function call:	`void InitTDC()`
Description:	The function `InitTDC` has to be used to initialise the starter kit and to synchronise the application software with the starter kit. The default settings are described in Table 24: structure TDC.
Parameter:	n/a
Return value:	n/a

Table 15: function InitTDC()

Function call:	`void RestorePinSettings()`
Description:	The function `RestorePinSettings()` loads the saved TDC structure program settings to the GUI an initialised the starter kit afterwards with this settings.
Parameter:	n/a
Return value:	n/a

Table 16: function RestorePinSettings()

Function call:	`void SavePinSettings()`
Description:	The function `SavePinSettings()` has to be used to save current program settings in the TDC structure.
Parameter:	n/a
Return value:	n/a

Table 17: function SavePinSettings()

Function call:	`void Wait(Int32 ms)`	
Description:	The function `Wait` can be used to stop the current software process and to wait a defined time `ms`	
Parameter:	`ms`	Time to wait in milliseconds
Return value:	n/a	

Table 18: function Wait()

Variable:	`Button AutoConnectButton`
Description:	This is the Connect/Disconnect Button of the GUI. It's useful to combine the initialisation of your special GUI elements and settings with this button.

Table 19: variable AutoConnectButton

Variable:	`label VersionLabel`
Description:	This label is partly predefined. It contains the version information of MSCSTK-LIB. You can use this element to mark your software version. It is displayed below the GUI title.

Table 20: variable VersionLabel

Variable:	`ProgressBar WorkingProgressBar`
Description:	The progress bar is used to show that a measurement program or others are running. This is done by increase the value of the progress bar.

Table 21: variable WorkingProgressBar

Variable:	`double CalClkPeriod`
Description:	This variable is predefined with `250` ns and corresponds with the used 4MHz oscillator on the starter kit board.

Table 22: variable CalClkPeriod

Variable:	`int GlobalTimeOutValue`
Description:	This variable is to modify the time out behaviour of the application. The default value is `10000` ms.

Table 23: variable GlobalTimeOutValue

Variable:	`TDCStruct TDC`
Description:	This structure stores all settings of all supported TDC starter kits. Additionally all selections made in the `OptionDialog` Form are stored in this structure. The following values are the predefined values, which can be always set by `InitTDC()`:

Value	Default	Description
Starter kit Connection status:		
`TDC.Connected:`	[2]	Connection status (boolean)
`TDC.NotInitialized:`	[2]	Initialisation status (boolean)
`TDC.Typ:`	[2]	TDC number (e.g. "`502`" = TDC502)
`TDC.UART:`	[2]	UART or USB connection (boolean)
`TDC.Name:`	[2]	TDC name string (e.g. "`TDC502`")
Starter kit hardware settings:		
`TDC.LED:`	False	Green Run-LED (on=true)
`TDC.OSC:`	False	Starter kit oscillator (on=true)
`TDC.SPI:`	False	Use SPI (TDC-MACH-3*0)
`TDC.ENSPI:`	False	Enable SPI (TDC-MACH-3*0)
`TDC.MODE0:`	True	MODE0 (TDC501)
`TDC.MSDI:`	False	MSDI (TDC501)
`TDC.NOEN:`	False	NOEN (TDC501)
`TDC.NSTA:`	False	NSTA (TDC501)
`TDC.NSTP:`	False	NSTP (TDC501)
`TDC.EN_STOP_A:`	False	Enable STOP_A (TDC502)
`TDC.EN_STOP_B:`	False	Enable STOP_B (TDC502)
`TDC.PINENA0:`	False	Enable channel 0 (TDC10000)
`TDC.PINENA1:`	False	Enable channel 1 (TDC10000)
OptionsDialog variables:		
`TDC.NumberOfMeasurements:`	10	OptionsDialog variable
`TDC.ChannelASelect:`	True	OptionsDialog variable (TDC10000, TDC502)
`TDC.ChannelBSelect:`	False	OptionsDialog variable (TDC10000, TDC502)
`TDC.Resolution:`	4	OptionsDialog variable (TDC502)
Clock divider variable:		
`TDC.CalDiv:`	4	Clock divider setting[3]

Table 24: structure TDC

[2] If connected with a starter kit, these variables get their starter kit specific value.

[3] This variable is only for the software and is not the same as the clock divider setting within the TDC.

4.4.4 Working with the `OptionDialog` form

Function call:	`void GenericOptionsDialog(bool Channel, bool Resolution, bool Measurements)`	
Description:	This function shows the `OptionDialog`. The dialog form has to be initialised using the parameters.	
Parameter:	`Channel`	Shows radio button box “Select channel” with selection between channel A and B, only one channel can be activated (TDC10000, TDC502)
	`Resolution`	Shows the radio button box “Select resolution” with 4 values: 1=half, 0=normal, 2=high, 4=smart (TDC502)
	`Measurements`	Shows a text input line to set the number of measurements
Return value:	n/a	
Example:	see figure 6: select options dialog `GenericOptionsDialog(true, true, true)`	

Table 25: function GenericOptionsDialog()

Function call:	`void ShowConfiguration(int Mode)`	
Description:	This function shows the configuration selected by the `OptionDialog` form and the selected measurement mode `Mode` in the protocol box.	
Parameter:	`Mode`	Shows the selected measurement mode `Mode`.
Return value:	n/a	

Example:	`ShowConfiguration(7);`
	
	`Your selected configuration:`
	`Channel(s): A`
	`Resolution: smart`
	`Measurements: 10`
	`Mode : 7`	

Table 26: function ShowConfiguration()

4.5 Implementing your program into the GUI

The delivered template file “UserButtons.cs” can be used to implement your own measurement application in the starter kit application software. This file initialises an own MSCSTK Form `MyMSCSTKForm` with 10 additional `UserFunctionButton[]` user function buttons. To initialise the buttons a `UserConnectButtonHandler` handler is started with the `AutoConnectButton` button click.

The `UserConnectButtonHandler` handler defines the button text according to the used starter kit if it is connected. Otherwise all user function buttons are set invisible.

If a user function button was clicked, the `UserFunctionButtonHandler` handler is called. According to the pressed user button and the used TDC a measurement function is called.

There are a number of predefined function calls for the user function buttons.

Example 3: Place a program on user function button 9

Add to the function `UserFunctionButtonHandler`:

```
case 9:
        {
        // Program control variable must be activated!
        RunProgram = true;

        // Title
        WriteText(Environment.NewLine + "My first user measurement program!", TitColor);

        // Output the connected TDC-Type and some settings...
        WriteText("Get settings:", InfColor, FontStyle.Underline);
        WriteText("Connected TDC from starter kit is " + TDC.Name + ".", InfColor);
        WriteText("The oscillator is " + (TDC.OSC ? "on" : "off") + ".", InfColor);

        // Switch on the oscillator
        if(! TDC.OSC)
                Cmd("OSC_EN");
        WriteText("Now the oscillator is " + (TDC.OSC ? "on" : "off") + ".", InfColor);

        // 10 times LED blink
        WriteText("Take a look at the blinking green LED...", InfColor);
        for (int myTempInt = 0; myTempInt < 10; myTempInt++)
        {
                if (TDC.LED)
                        Cmd("LED_OFF");
                else
                        Cmd("LED_ON");
                Wait(100); // wait for 0.5s to see a blinking LED!
        }

        // Switch the LED on finally
        if (!TDC.LED)
                Cmd("LED_ON");
        WriteText("Now the green LED is " + (TDC.LED ? "on" : "off") + ".", InfColor);

        // Finish
        break;
}
```

Add to the function `UserConnectButtonHandler` under the if construct `if (TDC.Connected)`:

```
// My user function button 9
UserFunctionButton[9].Text = "My 1st function";
```

If the compiled application runs, press the Connect button and afterwards the user function button 9. You should see a blinking green LED on the starter kit and the following text in the protocol box:

Example 4: button 9 output in protocol box

```
My first user measurement program!
Get settings:
Connected TDC from starter kit is TDC-MACH 300.
The oscillator is off.
Now the oscillator is on.
Take a look at the blinking green LED...
Now the green LED is on.
Program execution was successful.
```

5 Appendix

5.1 Specification

5.1.1 Common

- 32 MHz M16C/60-microprocessor.
- Time-to-Digital Converter TDC-Mach300.
- Control pins of the TDC are configurable via microprocessor.
- 2 SMB measurement-inputs for high precision time difference measurements.
- 4 MHz oscillator, used as TDC-calibration clock (30ppm).
- Red LED for power-on indication.
- Green LED for indication of correct microprocessor operation.
- Communication between PC and evaluation board via USB 1.1.

5.1.2 Minimum system requirements

- PC with USB 1.1 interface.
- Operating system: Windows XP, Windows Vista, Windows 7 with .NET-framework 3.5 or later [4]
- Measurement inputs with 3,3V/5V compatible ports for start and stop.

5.1.3 Power supply

- Standard universal power supply connected to X201; min. 9V up to max. 12 V AC/DC.
- Maximum current consumption of the board: approx. 200 mA.

To guarantee a high degree of signal integrity the board provides two independent supply voltages, one for the TDC chip and one for the microprocessor. So the power supply of the TDC chip is not affected and disturbed by other components. The TDC's supply voltage is adjustable via the potentiometer R209 within the voltage range of 2,7V up to 5,5V. The supply voltage of the microprocessor and all other components is adjustable via the potentiometer R225 within the voltage range of 3,0V up to 5,5V.

Caution: The values of the two supply voltages may not differ from each other more than 0,5V!

5.1.4 Reset

When the board is connected to the power supply, the board's reset chip runs a power-on reset cycle. This resets the microprocessor and the TDC chip as well. If the supply voltage of the microprocessor drops to values less than 3,0V a reset may occur, too.

[4] The application software was tested using Windows XP Professional Edition 32-bit and Windows 7 Professional 64-bit

5.2 Layout diagrams / schematics

Figure 7 shows the top view of the board's layout diagram and Figure 8 and Figure 9 show its schematics. The board's connector X301 is connected to the TDC's pin START, X303 is connected to STOP.

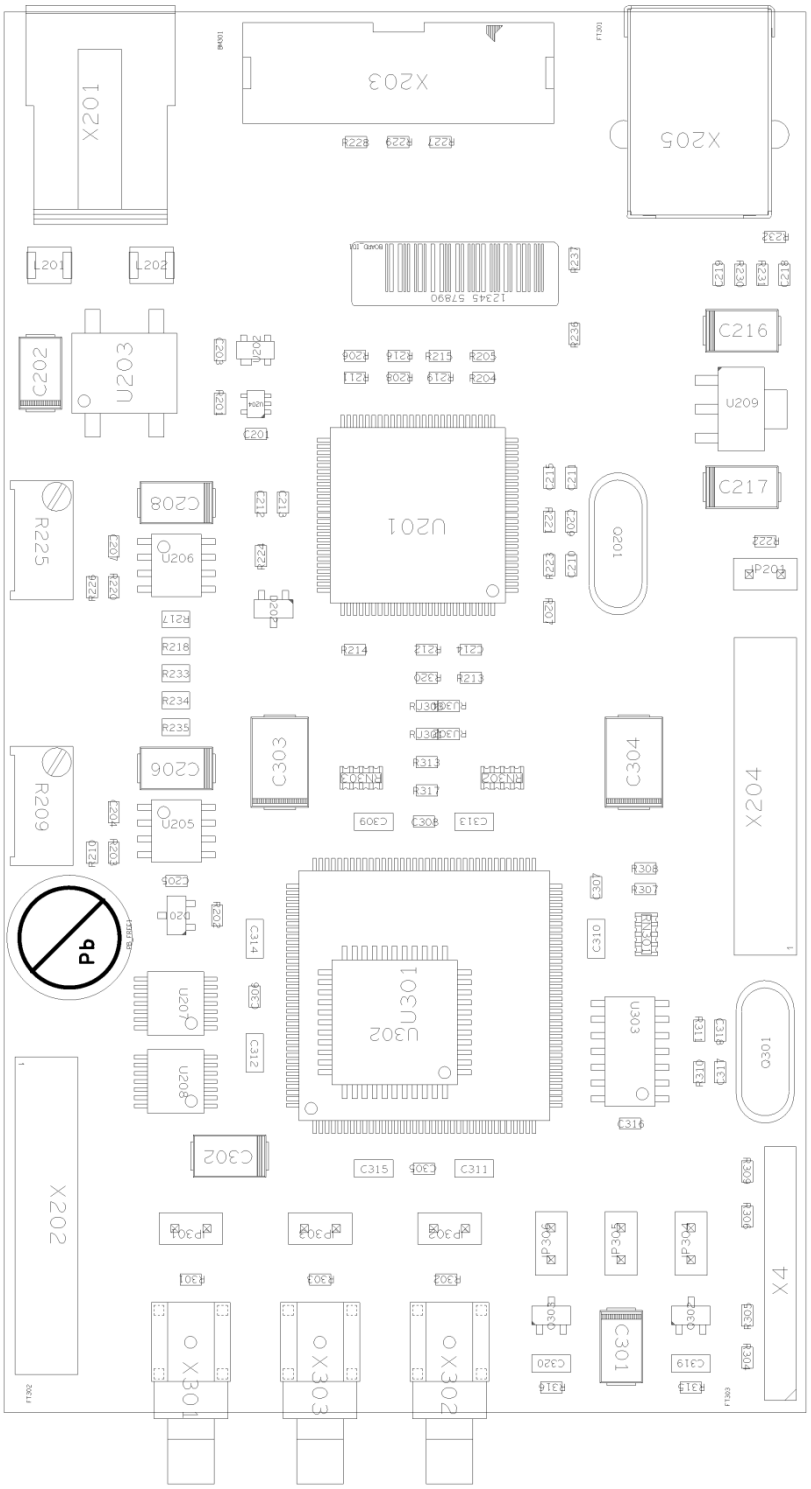


Figure 7: Layout diagram TDC-Mach300 starter kit (top view)

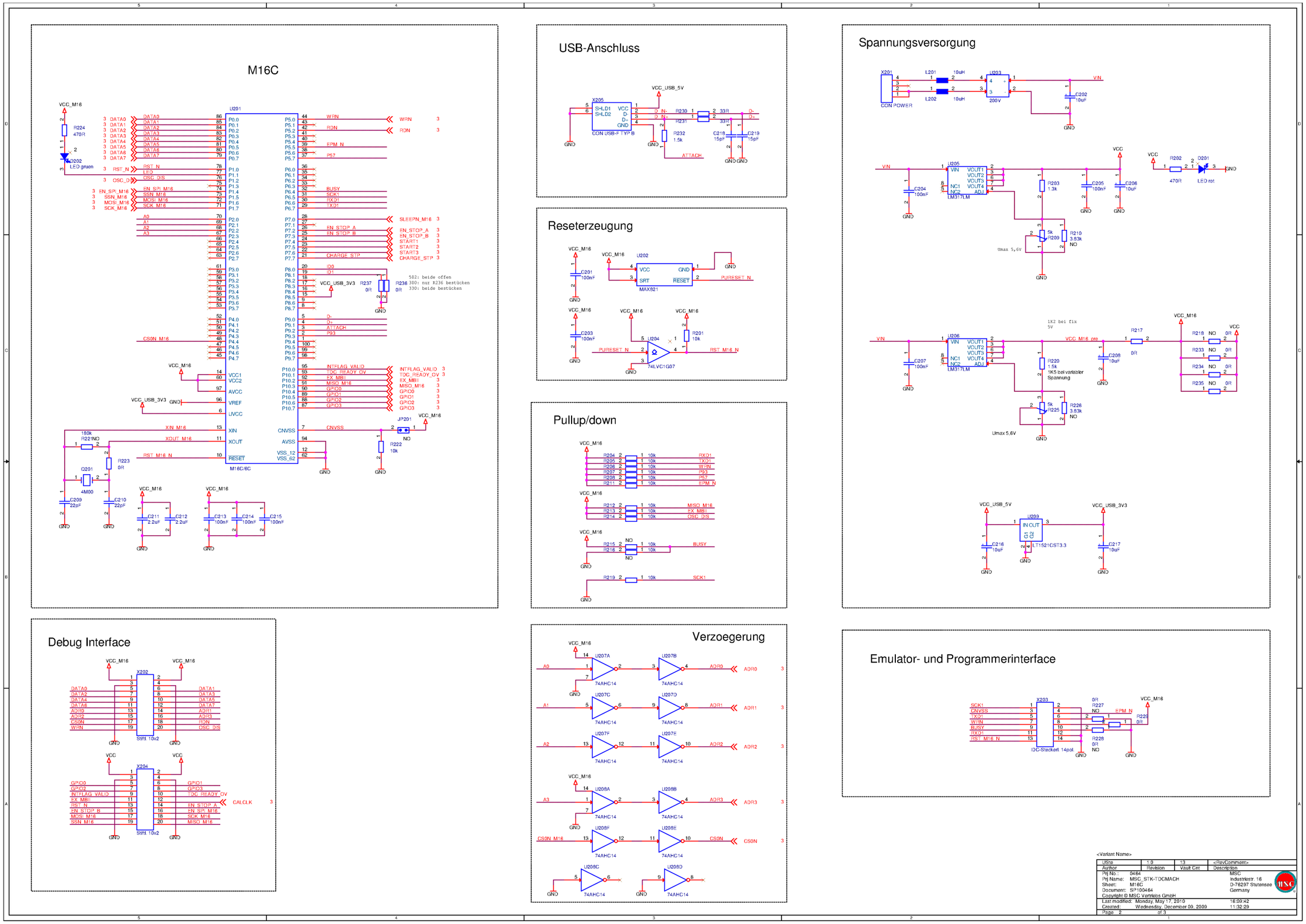


Figure 8: Schematic of the TDC-Mach300 starter kit (part 1)

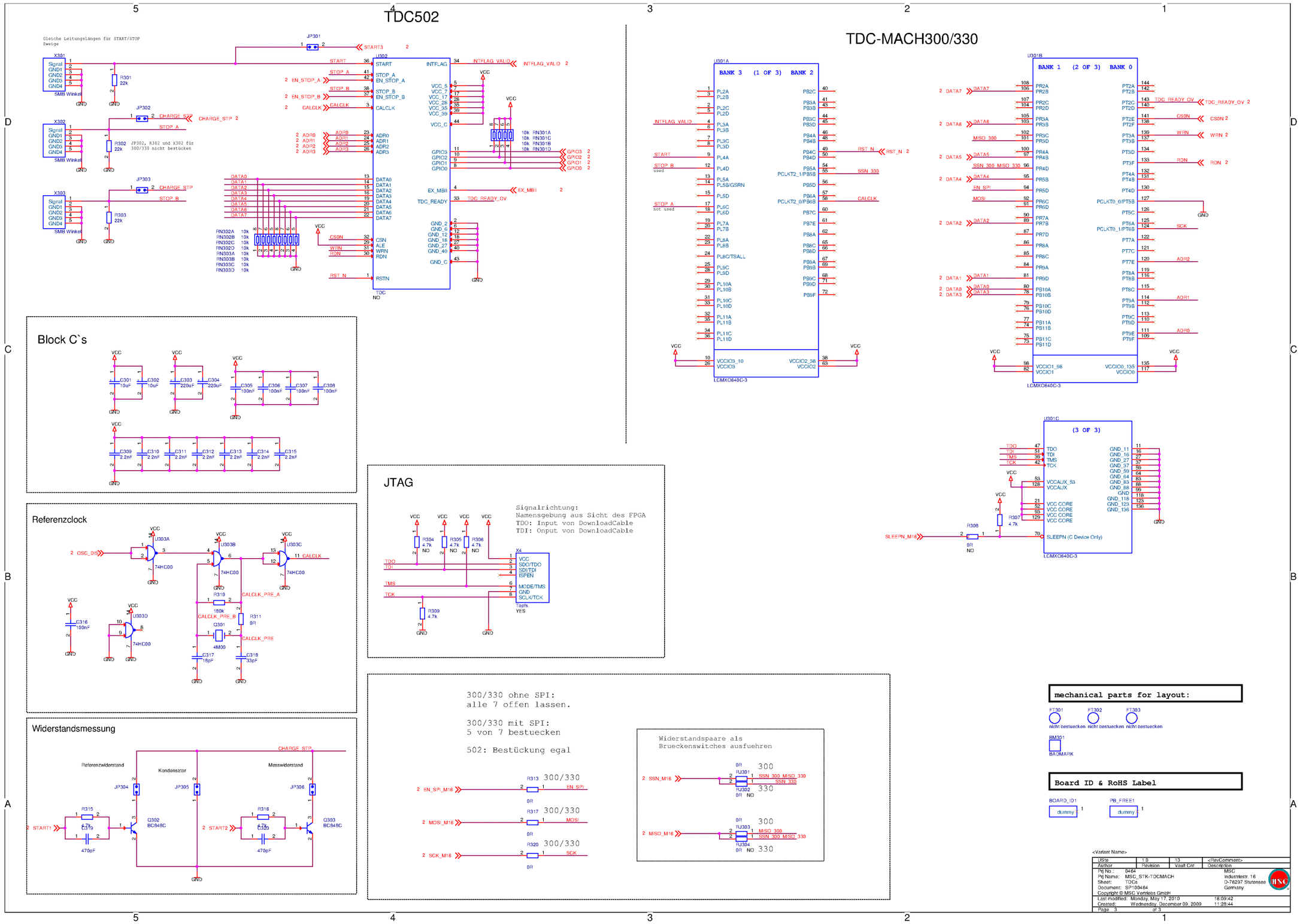


Figure 9: Schematic of the TDC-Mach300 starter kit (part 2)